S/N 10/731,597 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Robert Little et al.	Examiner:	Amelia L. Rutledge
Application No.:	10/731,597	Group Art Unit:	2176
Filed:	December 9, 2003	Docket No.:	60001.0270US01
Title:	PROGRAMMABLE OBJECT MODEL FOR NAMESPACE OR SCHEMA LIBRARY SUPPORT IN A SOFTWARE APPLICATION		

DECLARATION UNDER 37 CFR 1.131

I, Marcin Sawicki, citizen of the United States of America and resident of the state of Washington, declare: "Exhibit 1" is an invention disclosure form that substantially sets forth the invention as claimed in the present patent application. For example, the second and third full paragraphs on page 1 describe the elements of claim 1 (which are also schematically represented in the figure on page 5). I authored the invention disclosure at least on or before November 11, 2002, which is the date documented in the invention disclosure form.

I, the undersigned inventor, declare that all statements herein made of my own knowledge are true and that all statements are made on information and belief and are believed to be true, and further that these statements are made with the knowledge that willful false statements and the like are punishable by fine and/or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

08/08/2006
Date

Marcin Sawicki
1806 Second Street
Kirkland, WA 98033

Exhibit 1

Microsoft Patent Pre-disclosure Document

Title of Invention: Programmable Object Model For Namespaces Library Support In an Application

Date: 11/11/02

Document Author(s): Marcin Sawicki (marcins@microsoft.com)

Introduction:
The goal of this invention is to provide an object-oriented programming model for the XML namespace library functionality of a software application (see patent disclosure ########). It is relevant to a software application that is capable of opening, representing, processing, manipulating, editing or outputting in any other way data or documents as XML. A programmable object model in general enables such an application to serve as a programming environment for programs (a.k.a. solutions) or as a tool used in other programs to process XML. This specific invention makes the namespace library feature of an application available and programmable to custom solutions. A namespace library is a mechanism for associating files (such as schemas, xslt transformations and others) and solutions with a particular namespace so that when the application encounters xml markup from a given namespace, it can quickly and seamlessly take advantage of all those additional files associated with that namespace without user intervention. Thanks to this invention, the XML namespace library functionality of an application can be taken advantage of programmatically by other programs written in any language (e.g. C, C++, C#, Visual Basic).

Programmers often wish to take advantage of an existing application's capabilities in their own programs or to customize the functionality of an application and make it more suitable for a specific set of users or actions. For example, a programmer working in the financial industry may wish to customize a word processing application for a user audience consisting of financial analysts editing financial reports. There exists huge interest in XML as a standard interchangeable data format for the financial industry as well as many other industries. An application capable of processing XML data and supporting the namespaces library functionality already provides a lot of value to its users. But unless it exposes its XML functionality via a programming model, programmers wishing to customize it further or to take advantage of it in their own solutions are out of luck. This invention solves that problem by exposing the functionality of a namespace library via an easy to program, intuitive and powerful object-oriented programming model.

Some of the programmatic capabilities of the object model related to this invention include:

- Ability to programmatically associate XML schemas with XML data via its namespace, and to detect and remove any existing associations
- Ability to programmatically associate XSLT transforms with XML data via its namespace, and to detect and remove any existing associations
- Ability to associate any other files or executable software with XML data via its namespace and to detect and remove any existing associations

Strategic Importance of Invention:
One of the most important selling points in Office 11 for corporate customers is what is commonly referred to as "Office as an XML Development Platform". XML namespace library support in Office is a key component of that and it is already enjoying huge interest internally as well as among external customers, journalists and competitors. What makes Office an XML development platform is its XML object model – the subject of this invention. Without it, Office would merely be an XML processor without the extensibility, programmability and customizability provided by the XML object model. As such, the XML object model in Office clearly gives Microsoft a competitive advantage as it is one of the first (if not the first) application suites to support such a rich XML object model.

Looked at from a traditional programmer's perspective, the object model is essentially a set of new API's exposing Office 11's XML functionality for product interoperability. It is related to the W3C's XML DOM standard, as implemented by Microsoft's MSXML parser. Virtually every word processing competitor is looking at XML these days and positioning their XML functionality as a solution platform will be key to many of them. Today, they

include Sun (StarOffice), Corel (WordPerfect, XMetal), ArborText (Epic), Adobe (FrameMaker) and many other software vendors.

Description of the Invention:
Below is an outline of the entire XML namespace library related object model as implemented in Microsoft Word 11 (the items in **boldface** are the new additions covered by this invention. The others are other objects that are shown here to provide context):

NAMESPACE LIBRARY

Application object

.XMLNamespaces property

ReadOnly pointer to an XMLNamespaces collection which represents the namespace library available to the application

XMLNamespaces collection

The top level object providing access to the XMLNamespace objects. It represents the namespace library. Each XMLNamespace object in the collection represents a single and unique namespace in the namespace library. The following are methods and properties of the object:

.Add() method

A method creating and adding to the collection a new XMLNamespace object. It is used to register a new namespace in the namespace library. It returns a new XMLNamespace object. It can accept the following parameters:

Path – pointer to the schema file for the namespace; can be a file path represented as a string
NamespaceURI – the URI of the namespace to be created; can be a text string
Alias – a text string representing an alternate (more user-friendly) name for the namespace that the programmer may specify
InstallForAllUsers – a flag indicating whether the new namespace in the namespace library should be available to all users of the computer or only the current user.

.Application property

A ReadOnly pointer to the Application object representing the application of this object model

.Count property

A ReadOnly property returning the number of registered namespaces in the namespace library. It is the same as the total number of XMLNamespace objects in the XMLNamespaces collection

.Creator property

A ReadOnly pointer to the creator of the object

.InstallManifest() method

A method for installing solution manifests that register namespaces in the namespace library. It can accept the following parameters:

Path – pointer to the manifest file for the manifest; can be a file path represented by a text string
InstallForAllUsers - a flag indicating whether the new namespaces installed the namespace library by the manifest should be available to all users of the computer or only the current user.

.Item() method

A method for accessing the invividual members of this collection using an numerical index or a search keyword. It can accept the following parameters:

Index – a number representing the position of the requested XMLNamespace object in the namespace library; it can also be a text string representing the alias or the URI of the requested namespace

.Parent property

A ReadOnly property returning the parent object of the collection. Typically, this returns a pointer to the application that the XMLNamespaces collection is accessed from.

XMLNamespace object

Object representing an individual namespace entry in the namespace library (and an individual item in the XMLNamespaces collection). The following are methods and properties of this object:

.Alias property

A property for controlling the alias the programmer associated with the namespace. It can support the following parameter:

AllUsers – a flag indicating whether the alias is available to all users or just the current one

.Application property

A ReadOnly pointer to the Application object representing the application of this object model

.AttachToDocument() method

A method for attaching the schema of the namespace represented by the object to the selected document. It supports the following parameters:

Document – a pointer to the document to which the schema is requested to be attached.

.Creator property

A ReadOnly pointer to the creator of the object

.DefaultTransform property

A property that points to the default XSLT transformation associated with this namespace. It can support the following parameter:

AllUsers – a flag indicating whether the default transformation setting should affect all users of the machine or only the current user

.Delete() method

A method for removing the XMLNamespace object from the collection and destroying it, effectively removing the namespace association represented by this object from the namespace library

.Location property

A ReadOnly property that controls the location of the schema associated with the namespace represented by the XMLNamespace object. It can support the following parameter:

AllUsers – a flag indicating whether the schema location setting should affect all users of the machine or only the current user

.Parent property

A ReadOnly property returning the parent object of the XMLNamespace object. Typically, this returns a pointer to the XMLNamespaces collection that the object is a member of.

.URI property
A ReadOnly property returning the URI of the namespace represented by the object

.XSLTransforms property
A ReadOnly pointer to the XSLTransforms collection representing XSLT transformations associated with the namespace represented by the object

XSLTransforms object
The top level object providing access to the XSLTransforms objects each of which represents a single and unique XSLT transform associated with a namespace in the namespace library. The following are methods and properties of the object:

.Add() method
A method for creating and adding to the collection a new XSLTransform object. It is used to associate a new XSLT transformation with a namespace in the namespace library. It returns a new XSLTransform object. It can accept the following parameters:

Location – pointer to the XSLT file; can be a file path represented as a text string
Alias – a text string representing an alternate (more user-friendly) name for the XSLT transformation that the programmer may specify
InstallForAllUsers – a flag indicating whether the new namespace in the namespace library should be available to all users of the computer or only the current user.

.Application property
A ReadOnly pointer to the Application object representing the application of this object model

.,Count property
A ReadOnly property returning the number of registered XSLT transforms for a given namespace in the namespace library. It is the same as the total number of XSLTransform objects in the XSLTransforms collection

.Creator property
A ReadOnly pointer to the creator of the object

.Item() method
A method for accessing the invividual members of this collection using an numerical index or a search keyword. It can accept the following parameters:

Index – a number representing the position of the requested XSLTransform object in the namespace library; it can also be a text string representing the alias of the requested XSL transform

.Parent property
A ReadOnly property returning the parent object of the collection. Typically, this returns a pointer to the application that the XSLTransforms collection is accessed from.

XSLTransform object
Object representing an XSLT transformation associated with a namespace in the namespace library

.Alias property
A property for controlling the alias the programmer associated with the XSLT transform in the namespace library. It can support the following parameter:

AllUsers – a flag indicating whether the alias is available to all users or just the current one

.Application property
A ReadOnly pointer to the Application object representing the application of this object model

.Creator property
A ReadOnly pointer to the creator of the object

.Delete() method
A method for removing the XSLTransform object from the collection and destroying it, effectively removing the association between the XSLT transoform and its namespace in the namespace library

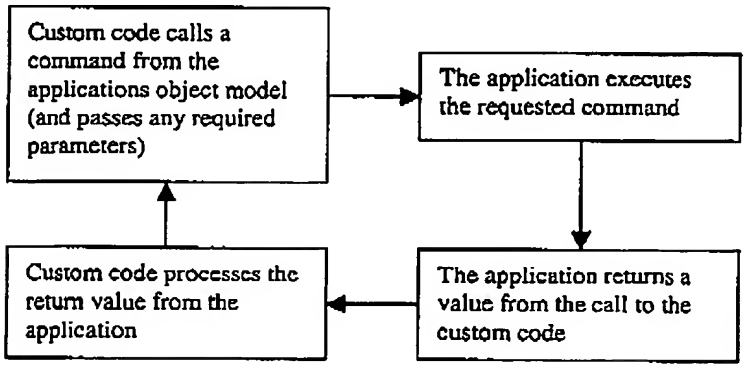
.Location property
A ReadOnly property that controls the location of the XSLT transform associated with the given namespace and represented by the XSLTransform object. It can support the following parameter:

AllUsers – a flag indicating whether the XSLT transform location setting should affect all users of the machine or only the current user

.Parent property
A ReadOnly property returning the parent object of the XSLTransform object. Typically, this returns a pointer to the XSLTransforms collection that the object is a member of.

Diagrams and Flow Charts:
[To support the description provided above, please include: (a) at least one block diagram showing the architecture of the system that implements your invention, and (b) at least one diagram illustrating the primary steps performed by your invention.]

The architecture of the system is just a typical computer with memory and CPU, input devices (keyboard, mouse), with word processing software running on it. It could be a tabletPC, a palmtop, a server, a terminal



Invention Timeline:
[Please provide the date that the solution was first conceived. Also, please provide the date that your invention was reduced to practice, and any prior or planned disclosure of this invention outside of Microsoft.]

Date of Conception: Mar. 2001
Date Reduced to Practice: Office 11 Beta 1

Prior/Planned Disclosure:

Beta 1 was shipped to NDA customers in **October 2002**
Parts of the Object Model were demoed and described publicly (non-NDA) for the first time during the XML Conference in Baltimore (**December 14, 2002**)

Additional Information:

- List any people that should be considered as an inventor and their role in developing this solution:

Brian Jones – main inventor, spec creator
Marcin Sawicki – main inventor, spec creator
Rob Little – developer, who invented some of the API in the object model
Mark Sunderland – developer, who implemented parts of the invention and helped modify aspects of the invention

- List any non-Microsoft inventors and their employers.

Ed Tharp – developer who implemented most of the invention and helped modify significantly various aspects of the invention; retired from Microsoft in 2002

- List any earlier, current or anticipated MS products that may use your invention:

Microsoft Word 11 implemented all of the invention
Microsoft Excel 11 implemented parts of the invention

- List and attach (or provide pointers to) any documents that provide additional information about your invention or the product to which it relates, including specifications, journal articles, slide presentations, test/performance results, etc.]

http://officenet/specs/XMLOM.xml

- List any other sources that would provide helpful background information or illustrate prior work of others in this area (including, e.g., journal articles, text books, product literature, products, and specifications):

Parts of the invention were inspired by the designs of XML object models in the MSXML parser (a Microsoft product) and in the W3C DOM standard (Document Object Model):
http://msdn.microsoft.com/library/en-us/xmlsdk/htm/dom_reference_2kdh.asp
http://www.w3.org/TR/DOM-Level-2-Core/

- List any non-Microsoft methods for solving this problem (e.g. java does something similar for Sun):

Any implementation of the W3C DOM in a competing word processor would be intended to solve this same problem. I'm not sure if Java, or WordPerfect, or XMetal actually do this, but it is conceivable they could.